My own Python onliner

This is the first step to my own outliner software. An outliner is nothing more than a text editor with additional functions for organizing your work. First and foremost there is nothing special, but still many programmers who also write texts for blogs use this kind of editor. I had used Dave Winer's OPMLEditor on Windows 10 a few weeks ago and was enthusiastic about the idea. It had quite a few functions that I liked. But there were some things I didn't find necessary for my personal work either. Unfortunately, the Outliner only ran on Apple or Windows. I could (maybe) port the software to Linux so I can use it, but unfortunately I can't do it in my spare time. Jörg Kantel, a well-known German blogger, brought RubyFrontier and TextMate to my attention. Unfortunately they are only available for Apple and the hardware is not available at the moment. So I had no other option but to develop something myself, even if it will cost a lot of time. In this developer diary I will record the most important steps of development and also link the sources I have worked with. This should help people who also play with the idea of developing an outliner or programming in Python.

I chose Python because it is well documented and used by many developers. So you can access a large number of packages, side projects and tutorials. With this language it is possible to develop complex software and to run it on Linux, Apple and Windows. For the user interface I use PyQt4 and for the development environment Linux Mint in the latest version. First I arranged the interface and made a plan of what I want to have implemented. This includes a text editor, an update function, a grabber for RSS and a small web browser. There are enough tutorials for everything and that won't be a problem that can't be implemente

First of all, it is important that the online user can access himself. So you have to develop a starter that manages the individual modules. I like the idea of Apple software, which inserts a bar on the desktop when opened and from there the individual windows are operated. So far I haven't found out how to implement it on Linux. One would have to be able to determine quasie like CSS that the individual (bar) window should always be shown at the top of the screen, has only a certain height and should always be 100% in width. I still have some research to do in this direction. So far this is shown as a small bar in the upper left corner. From there you can open the text editor and the other modules.

Since I want to make this as understandable as possible for users, there is a single file for each function or software (or script). These files are the modules you can work with. It is comparable to Minecraft, which also implemented this idea. For example, if I use the text editor, I can look at the source code in the `textEdit.py` file. I want users to start customizing the outliner to their own needs. Personally I can also work better with it because I like the concept of web development. You have an `index.html` file and all other files reference it. It brings a very good structure into the project when all files for the text editor are in the corresponding folder. When I look for the Urls file for the RSS Grabber, I find it in the RSS folder etc.

I had already developed most of the modules as terminal versions. For example, the RSS Grabber has been developed with BeatifulSoup4 and its own color class. The grabber does nothing more than read out my rss feed and has been developed according to the unix philosophy. I later decided to get away from the terminal and use a real user interface. It not only looks better, but is also easier for users to understand if you have the colorful buttons to press with. I also want to develop for a larger target group than the hard Linux programmers club. Since this is still a private project, I will not publish any source code yet and will wait until it has been implemented reasonably properly. I have no interest in having to talk to people because this and that is broken or not yet working. I can do that later with a real public version. Also I have no idea under which license this should run. It is planned that this will run under an open source version. But I'm not sure which one yet, because there are a lot of them.

textEdit Basic Functions

After the basic features of the onliner have been developed, the first module could be further developed. textEdit will be the module for editing texts. No matter what you work with, you will always deal with regular text first. Whether it is the `readme.txt`, a `.md` file or `.html` is what you want to edit. All can be opened with a regular text editor. Since I work minimalistically and all elements concern the design (among them icons and other pictures) the basic functions did not take much time.

The File Menu item has been created. It contains the subitems new file, open file, save file and preview file (we will discuss this function in more detail in a moment). Although there are an incredible number of editors on the Internet, these simple features are crucial to how well our software works. You don't need any additional functions and no unique selling point if you simply deliver a good product. For example, when I am on the Linux terminal road, I usually use nano for short or small changes. Vim is brilliant (and emacs too), but also quite complex and the learning courses are more strenuous. I use Vim when I want to develop fancy software and need a lot more technology than I could get it from the nano editor. Both have advantages and disadvantages. At the beginning of the project I just wanted to integrate a Python editor into the onliner, but then I

left it. Most of them have already implemented far too much for me personally and prevent me from working.

To be able to print text files in pdf or Postscript the QPrintPreviewDialogDialog class has been added. I was inspired by the Building a text editor with PyQt. Since everything has already been prepared, there is nothing more to do. Also something you only have to learn when you develop a project. You don't always have to reinvent the wheel, but you should already know how to install a wheel and why it works the way it was designed. Many dumb questions on the Internet arise from the working technique of copying fragments of source code into his code and then wondering why it doesn't work properly.

Almost every software used for editing texts also offers these functions via key combinations. For example, Ctrl+C and Ctrl+V are already fixed in almost every operating system. Nevertheless, they are still built into software when you are on an exotic OS. Since I always like to be on the safe side, I looked at the rough structure in a forum and then read the rest in the QPlainTextEdit Class Reference. Some people also prefer to use the mouse and want to be able to click on it via a visible menu. With the current solution we have made both sides happy. In addition, if we expand the software, we can build on the work we have done so far. So all basic functions are implemented and in the next steps we can focus on the fancy functions that enhance everything.

Fullscreen and rss.xml formatting

This week I worked a few hours on a sidekick project. I had the idea to write scripts in Bash that create my rss.xml entries. These should be formatted immediately so that I can take care of the important things. I had simply started it because I had seen this simple blog system on the Internet and implemented it in a similar way. Today I continued programming on the PythonOnliner (I still didn't have a real name). First I implemented a function for fullscreen. Nothing special and a setup of two minutes. After that I wanted to do something, but with the best will in the world I couldn't get any further. What do you do when you can't see the forest for the trees and can't define the problem? That's right! You put it in the corner and do something else.

My basic idea (why I started the PythonOnliner project at all) was to make my work with the rss.xml file easier. So far I have done all this manually, which simply takes far too long. Only stupid people make so much work and effort. So I created a new menu item. In many text editors there is the function to include certain text blocks in a text. Couldn't I do something like that for my onliner? After about two hours I had the most important functions built in and can now also insert them into the text by key combination.

This allows me to create a completely new rss.xml file in four clicks. `Ctrl+N` new file, `Ctrl+F` Fullscreen, `Ctrl+B` insert body, `Ctrl+E` insert entry. Since people should later hack the source code themselves and change it according to their wishes, it is all programmed in a self-explanatory way. That's the quickest way for people to learn. I taught myself HTML because I opened the help files on a Windows 95 computer in a text editor. Eventually I started to edit them and was able to explain the source code better myself in small steps. At that time I didn't have an internet connection (and couldn't have afforded it in terms of price). In any case, the PythonOnliner should also be built in this way.

All in all, this can be expanded further. For example, if you prefer to create a blog article in Markdown for your static website. But you can also install a whole framework. It would cost a lot of preparatory work, but should be technically possible. Only then it will be difficult to accommodate all functions as key combinations, but this would be sufficient as a click-paste solution.

Alligator 0.01 (Python Onliner Software)

In the last two months I really had a lot to do and could not care much about the Python onliner. I did that now before the project was completely forgotten. I added new shortcuts for pictures, paragraphs and links so you can use them on your RSS blog. So far I only use functions that I need myself or use a lot. I also changed the upper and lower case and fixed many spelling errors. The source code has been made a little cleaner in many places. Also there is now an about page and if you click on the link below the default browser will open. I had chosen the name *Alligator* because it was running on a different monitor in a nature documentary. I downloaded the matching logo from Open Clip Art. I borrowed the web browser here and will completely rework it in the next weeks. That's just a fill-in now.

I wanted to implement a lot more (that's what I always want) and I miscalculated the effort. My roommate is of the opinion that I should rather publish in small upgrades, so that the frustration does not increase so much with me if I do not get on at a place. I think he's right, and I should keep it that way. For example, the web browser can't display formatted xml files yet, line numbers are missing and I didn't like the previous syntaxhighliting solutions. I'm going to set up a complete subpage with the help information for the Alligator 0.01 first and then make everything available for download.